SLUG NEWSLETTER

sinclair LOUISVILLE USERS GROUP

SEPTEMBER '87

The Sinclair Louisville User Group meeting came to order on Sunday, August 9, 1987 at 2:15 PM at the Government Center on Outer Loop road.

If you have articles or programs that are to be included in the newsletter, please give them to Shawn Merrick. He will then edit ,if necessary, and then pass them on to our publisher, Jim Dodrill. This procedure was determined to be, at our last meeting, the best path to follow for submission of articles or programs.

The raffle tickets should all be in for the drawing today. The prize is a TS 1000 with 16K rampack, soft rubber keyboard, and the donator claims it also has "an enhanced operating system". Only the winner will know for sure what that means. Drawing will be held on Sunday, September 13, 1987.

It has been mentioned that more participation in the meetings is necessary to keep this club going. That means ALL members , not just a select few, need to try and participate. The summer is almost over, and the winter indoor sport of computing is just around the corner, so lets see more T/S and ADAM users at the meetings!

We were visited by John Nygen at last meeting. John is presently away at school, but is a very active member when he is home. Anybody who has ever used the PIXFIX program, can appreciate the talents and knowledge John has added to the club.

Jim Pitts continued with his series of machine code classes with an explanation of the internal registers of the Z-80 microprocessor and how to use them within a machine code program. This is a very informative and interesting part of our meetings, and should help anyone interested in Machine Language programming.

After Jim's class, the meeting was officially ajourned at 4:15 PM. Give it your best shot and try to make the meeting next month on Sunday, October 11, 1987. We would really like to see you there.

Bob Crecco, Secretary

President	JIM PITTS	Editor	SHAWN MERRICK
Secretary	BOB CRECCO	Librarian	ED RADTKE
Treasurer	JOHN HAMILTION	Public Realtions	RAY MANLEY

Coleco ADAM Review #1

As computer prices have dropped for the past 3 years, it has become affordable to own more than one home computer.
If you are considering a computer purchase, you might really benefit from the ADAM system. Owners of the TS 1000 who program in machine language will be right at home with ADAM's Z80A CPU. The CP/M operating system is readily available providing a larger software selection.

Brief Description:

Price Range: $100-200 (used, or kit/expansion)

Note: ADAM is packaged as an entire hardware and software system. It is available as an add-on to the ColecoVision video game or a stand-alone unit with the ColecoVision game port built-in.

Hardware:

Keyboard- Detached, 75 full-stroke keys including 20 function keys. Excellent feel!

Memory Console- Z80A CPU (4 Mhz) with 80K RAM (64K+16K Video RAM). A master 6801 chip controls 6801s in peripherals, allowing ADAM to do 2 things at one time! TMS 9918A graphics chip has 16 colors (and used in TI99/4A computer). 1 Digital DataPack drive is installed with room for a 2nd (@ approx $20). Cartridge port plays all ColecoVision games from Coleco and most from other makers. 3 different expansion slots allow for internal 300 baud modem ($30), 64K RAM expander ($45, 144K RAM total), and a future hardware project. Bus connector on side works with Atari 2600 game adapter or Auto-Dialer (w/address book). Up to 2 disk drives can be connected to the modular AdamNet plug.

Digital DataPacks look like audio cassettes, but have special alignment holes drilled in them to prevent mix-ups. They are completely automatic (controlled by the computer) and just a little slower than disks. The precision +/- .002" tape alignment might be the cause of temporary read errors, but a quick retry usually works. On rare occassions, the tape might switch directions so quickly for a retry that it gets tangled around the hub inside the shell. If this happens, much of the data can be recovered by unscrewing the shell and untangling the tape. DDPs hold 256K and provide a built-in standard media for ADAMs while beating regular audio tapes by a long shot. However, the ADAM disk drive (available seperately) makes ADAM even more efficient.

Printer- Letter quality daisy wheel uses multistrike carbon ribbons. It's a little slow at 10 characters per second, and slightly louder than a typewriter, but the print quality is great.
This review was printed on the ADAM.

Power Supply- Only 1 power supply is needed for the unexpanded system. It is built into the printer, and slightly awkward since the power switch is on the back. Seperate power supplies are available as surplus and can be used without the printer.

Joysticks- 2 included with numeric membrane keypad. Many different game controllers (roller, super, steering wheel) are also available.

Software:

SmartWRITER Word Processor- built into ROM and loads when a programmed tape or disk are not in the drives. Originally sets up in typewriter mode; press WP/ESC key for word processor. With the exception of a few quirks and non-existant features, it is very easy to use and very productive. 6 Smart Keys (function keys) provide a menu-key system. Save files to DDP1, 2, or disk 1.

SmartBASIC- This imitation of AppleSoft BASIC loads from DDP and runs many Apple programs that don't use hardware-specific memory locations (PEEK or POKE). As easy to use as any other BASIC.

Super Buck Rogers- This DDP shows off ADAM/ColecoVision game features, including the "play one screen while another is loading." One of many quality arcade games for ADAM.

Blank DDP- preformatted, can be duplicated onto ordinary cassettes with dubbing equipment by drilling alignment holes in each.

That is just an introduction of the ADAM's features. I think it is quite a bit for the money! I'll be reviewing some of the best software and hardware available in future issues. If you have any comments or need some ADAM help, please write me in care of this newsletter. Thank you.

Shawn Merrick

If you have a modem, you can check a computers certification status by calling the FCC's Public Access Link at 301-725-1072, 24 hours a day, seven days a week. The system is memu-driven, has help functions, and is configured to run at 300 or 1200 baud. There is a five minute limit on calls made between 8am and 8pm; after that the limit is 15 minutes. The PAL computer idenifies products by the number on their FCC certification stickers.

****

OCTOBER

SU	MO	TU	WE	TH	FR	SA
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

— SINC - LINK —

Sound Experimentor

Rob Miles

Unlike most home computers the QL has a very limited noise making capability. What there is can only be dragged out by using the enigmatic BEEP command. This has 8 arguments, all of which are obscure. The only way to produce something vaguely melodic is by trial and error. The following quickie allows you to do exactly this.

```
100 REMark **** QL User -Beep experiment
110 REMark ****  Rob Miles 1985
120 duration=5000:pitch=100:pitch_2=0:grad_x=0:gra
d_y=0:wrap=0:fuzzy=0:random=0
130 REPeat sounder
140 CLS
150 set "Duration (-32768..32767) ?",duration,2
160 set "Pitch (0..255) ?",pitch,4
170 set "Pitch_2 (0..255) ?",pitch_2,6
180 set "Grad_x (-32768..15) ?",grad_x,8
190 set "Grad_y (-8..7) ?",grad_y,10
200 set "Wrap (0..32767) ?",wrap,12
210 set "Fuzzy (0..15) ?",fuzzy,14
220 set "Random (0..15) ?",random,16
230 AT 18,0:PRINT "Beep ";duration;",";pitch;",";p
itch_2;",";grad_x;",";grad_y;",";wrap;",";fuzzy;",
";random
240 BEEP duration,pitch,pitch_2,grad_x,grad_y,wrap
,fuzzy,random
250 dum$=INKEY$(-1)
260 END REPeat sounder
270 DEFine PROCedure set (name$,var,position)
280 LOCal buf$
290 AT position,0:PRINT name$;" ";var
300 AT position,0:PRINT name$;" ";
310 INPUT buf$
320 IF buf$<>"" THEN var=buf$
330 AT position,0:PRINT name$;" ";var;"
340 END DEFine set
```

CATS
CAPITAL AREA TIMEX/SINCLAIR USERS GROUP
NEWSLETTER

DRESS UP YOUR GAME PROGRAMS WITH A SPLASH OF COLOR

by Lamont Downs

Here's a short program that you can merge with almost any game-type program and which enables you to change the paper color of the entire screen in a flash WITHOUT AFFECTING ANYTHING ELSE ON THE SCREEN. For example, if you've got a spaceship traveling across a black sky you can cause the entire sky to flash red for an instant and then back to black without erasing anything or changing any of the ink colors - and do it without having to remember any complicated USRs or even line numbers!

To use, first type in the program below and SAVE it. Then LOAD (or type in) the program you want to "decorate." Be sure the program does not have a line 1 or any lines from 9970 to 9989 (if it does, relocate them). Then MERGE the program below.

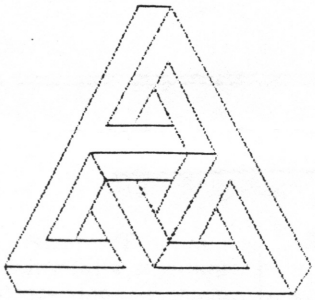
Now all you have to do to change paper color anywhere in the program is to insert the command GOSUB followed by the first three letters of the color (GO SUB blu, GO SUB gre, etc.). For example, if you have a blue sky background, you can light up the sky with a red flash with the following program line:

GO SUB red:PAUSE 2:GO SUB blu

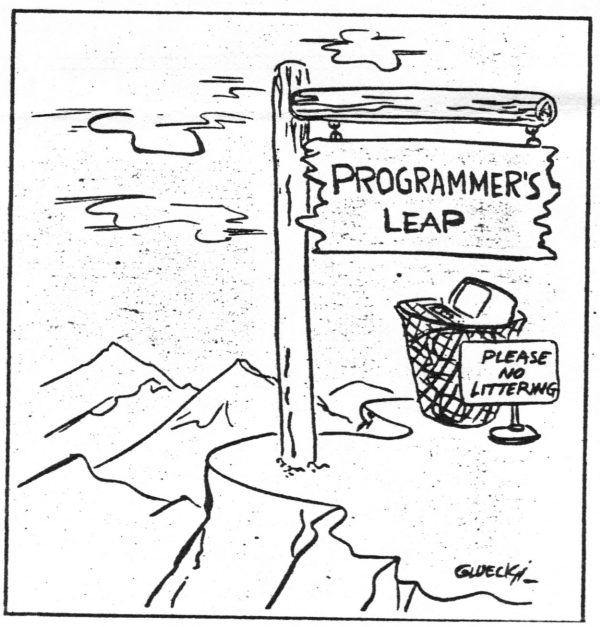
You can create dandy explosions by using sequences of colors (red-green-yellow-green-red) etc.). Some moniters will "tear" if colors change too quickly, so you may need to experiment with the length of the pauses for best effect. And don't forget you can throw in sound effects using the SOUND command instead of the PAUSE.

One last warning: until line 1 is executed the machine code for changing color is not loaded into memory and the program will crash if you try to change colors, so if you're starting the program somewhere other than line 1 be sure that you take this into account.

```
  1 CLEAR 65279: GO SUB 9970
9970 RESTORE
9972 FOR n=65280 TO 65304: READ
x: POKE n,x: NEXT n
9974 LET bla=9980: LET blu=9981:
 LET red=9982: LET mag=9983: LET
 gre=9984: LET cya=9985: LET yel
=9986: LET whi=9987
9976 RETURN
9978 DATA 1,192,2,33,0,88,126,20
3,0,203,0,203,0,119,35,62,0,11,1
85,32,241,184,32,238,201
9980 POKE 65288,159: POKE 65290,
167: POKE 65292,175: LET rand=US
R 65280: RETURN
9981 POKE 65288,223: POKE 65290,
167: POKE 65292,175: LET rand=US
R 65280: RETURN
9982 POKE 65288,159: POKE 65290,
231: POKE 65292,175: LET rand=US
R 65280: RETURN
9983 POKE 65288,223: POKE 65290,
231: POKE 65292,175: LET rand=US
R 65280: RETURN
9984 POKE 65288,159: POKE 65290,
167: POKE 65292,239: LET rand=US
R 65280: RETURN
9985 POKE 65288,223: POKE 65290,
167: POKE 65292,239: LET rand=US
R 65280: RETURN
9986 POKE 65288,159: POKE 65290,
231: POKE 65292,239: LET rand=US
R 65280: RETURN
9987 POKE 65288,223: POKE 65290,
231: POKE 65292,239: LET rand=US
R 65280: RETURN
```



```
  10 REM  By Frank Souldin, Fort
 Worth T/S Users Group
  15 REM Reprinted from the Jan/
87 issue of "The Data Expansion"
-- newsletter of the Fort Worth
T/S Users Group
  20 REM ----for TS 2068 machine
  30 LET d=200: LET p=160: CLS
  40 RESTORE p: READ a,b: BEEP .
05,34: PLOT a,b: LET p=p+10: RES
TORE d
  50 FOR n=1 TO 18: LET b=INT (R
ND*20)+15: READ x: READ y: DRAW
x,y: BEEP .05,b
  60 NEXT n: IF p=140 THEN GO TO
 80
  70 LET d=d+20: GO TO 40
  80 RESTORE 300: PRINT #0;AT 1,
10;"Press Any Key": PAUSE 0: FOR
 y=4 TO 26: READ c
  90 PRINT #0;AT 1,y;CHR$ c: BEE
P .05,x: LET x=x+1: NEXT y: PAUS
E 0: GO TO 30
 100 DATA 112,102
 110 DATA 20,18
 120 DATA 169,85
 130 DATA 224,3
 200 DATA 14,21,14,-22,-48,0,46,
70,100,-152
 210 DATA -13,-16,-197,0,-9,15,9
6,153,22,0
 220 DATA 85,0,-25,34,-11,-18,24
,0,-45,0
 230 DATA 32,51,70,0,-24,38,10,1
6,34,-54
 240 DATA -19,0,-11,-16,-48,0,-1
1,-17,11,17
 250 DATA -10,15,45,-66,67,0,-23
,36,10,16
 260 DATA -44,67,-23,-37,24,0,-4
5,0,35,51
 270 DATA -45,-66,11,16,-24,35,1
4,-20,12,20
 300 DATA 84,72,69,32,73,77,80,7
9,83,83,73,66
 310 DATA 76,69,32,84,82,73,65,7
8,71,76,69
```



UNLEASH THE POWER OF EXTENDED COLOR MODE ON YOUR T/S 2068!!

EXTENDED PAINT gives you eight times the normal color resolution. Over 25 unique functions. Joystick control, menu-driven. Extensive printer support, including hires greyscale screen dumps to T/S 2040 and Epson compatibles, as well as FULL COLOR hires dumps to Canon color ink-jet and compatibles. Supports AERCO Centronics interface. Cassette, with 24 page manual. $19.95 pp. from:

Dave Franson
3534A E. Squire Ave.
Cudahy, WI 53110

C.A.T.U.G.
NITE-TIMES NEWS

CHICAGO AREA TIMEX USER GROUP

IN DEFENSE OF PUBLIC DOMAIN SOFTWARE

By Pete Fischer

A friend of mine remarked to me the other day, "Why are you always messing with that PD (public domain) software? It's so CRUDE!"

I conceded how a lot of the PD stuff WAS, in fact, crude (though not all), but that its beauty was in its simplicity.

A commercial programmer goes to great lengths to conceal what he has done. His livelihood depends on getting remunerationtion for many hours of writing this code. The last thing he is likely to do is "give away" all that hard work. So he compiles it and saves the "SOURCE CODE" in his own files, selling only the "OBJECT CODE".

A PD program, however, is usually "open-architecture". This is primarily because the author didn't see any point in concealing it, but it may also be due to the fact that he's proud of his work and would be flattered if you copied it.

This is not to say that all PD programs are "open". Some are concealed, just like commercial ones and for similar reasons. The author may be pleased to let the computing public USE the program, but may be quite protective of the METHODS employed. Such is the case with some PD terminal programs and BBS programs in CP/M.

-=Learning to Program=-

The way I learned BASIC the MOST and the FASTEST was by simply "typing in" programs from magazines. Not the Machine Code kind which have numerous DATA statements, but the pure BASIC ones. I then tried various modifications to suit my particular applications. As I worked, I ran across new Keywords I'd never understood before and was forced to look them up. This seems to me a much more interesting way of learning than simply the dry memorization of one command after another.

In addition, you get to view other levels of organization, not merely the individual command, but a whole subroutine as it works together and the interaction of the various subroutines. This is a key point. In the process of typing in and/or modifying a program that you got out of the User Group library, you get to see every size of unit. You see 1) an individual command 2) a loop, 3) a subroutine, 4) the "business" part of the program (vs. the user-friendly front end), 5) the Graphics code, 6) the Instructions or DOCs and 7) the Program as a whole package.

But as long as there are computer users, there will be those who REFUSE to learn how to program. So be it. They will always pay TOP DOLLAR to get someone else to do those simple customizations that the rest of us take for granted. But I honestly feel that Timexers, in general, tend to learn programming early. This is due, in part, to the general lack of "shrink-wrapped" software, and part due to the EASE of programming that Sinclair computers embody.

Another advantage of PD software is that it sometimes does things that NO commercial package will do. Such as the Schoenwetter Printer patch, The TINYBOARD BBS (for nearly a year), for about 5 months, Mike DI RIENZO's RLE decoder was the way to go, for about 5 months Rich Moldovan had the only Terminal program for the QL which used XMODEM, and most recently, the only known RAMDISK for the 2068 was written by Jim Schowalter (525 Reed Av./Akron,OH 44301).

PD software evolved in the early days of computers, when mainframe programmers swapped their personal versions of Star Trek. But as the Micro Revolution began, there were those people who believed that ALL software should be free. (Part of an elaborate rationalization for "Why I pirate software") But they took their philosophy a step further. They proceeded to write some TRUELY EXCELLENT prorgams, and then distributed them far and wide. There are about 50,000 PD titles in CP/M. I get a chuckle, on occasion, when I read in an old T/S Magazine about a program that "Comes complete with a 9 page manual!". I have found many PD prorgams which come with a 100 page DOC file to explain it. Free, all free! But don't get me wrong, I'm very grateful for ALL the software developers who support my machines, and I'm NOT complaining about their charging for it. What I AM saying is that it's fairly difficult to "Over-Document" a piece of software.

Returning to the subject of "Type-in" programs. Where do I get such programs? Well, I sat down with my immediate collection of publications and paged through them all to count the Type-ins. I was pleased to discover that I own over 500 for my 3 computers (TS-1000, TS-2068 & QL). See the PD Chart for more details. I only had significant samples of 10 publications. If someone had all the newsletters to search, I'm certain they could round up quite a few more. I will guess that there are between 2-3,000 such type-ins in the obvious publications. There would also be more in the less obvious ones, like COMPUTER + VIDEO GAMES or YOUR COMPUTER (Both published in the UK, both carry type-ins for Sinclairs on a sporadic basis). I had no samples of these.

All these programs represent many man-years of time to input. Do I plan to type them all in? Not likely. That's where belonging to a User Group comes in handy. I can type in some, bring them in for the library and other members can type in others and we can swap. Ideally, this is done in a bit more organized fashion so that no two people end up typing in the same program. Organization is never perfect, but we can always try.

Now, I live way out here in Arizona, and I haven't really belonged to any national User Groups until fairly recently. But from here, I observe this: The national consciousness of and careful collection of T/S PD software has taken a dramatic leap in the last year. I currently belong to the Long Island Sinclair/Timex group (L.I.S.T.), and I really like their approach. Most national groups are happy to send you a newsletter (if you pay dues), but reserve their library priviledges for local members. LIST goes further, they have

IN DEFENSE OF PUBLIC DOMAIN SOFTWARE

tapes of PD software which they send to their "Mail Order" members as well. You can't order an individual program, you get an entire tape or nothing at all. You can get a tape by first becoming a member ($15/yr). If you have a program to donate, it costs you $3 plus a quality tape. If not, it costs you $6. The recent LIST tape #6.0 had 63 files on it. Not shabby for $3! But who do they con into spending their nights and weekends to MAKE these tapes? Simple, they bought a mono cassette duping recorder from Sears for $60 using the treasury money. Now our carefree librarian simply pops in two tapes, sets the machine going and wanders off to eat dinner. Whaddya think folks?

Nearly all the major User Groups have libraries, none (that I know of) have this excellent "Mail-order" policy of LIST. Contact them at:P.O.Box 438/Centerport,NY 11721-0438.

By my count, there are upwards of 100 User Groups here in North America. That represents a considerable pool of "typers". Suppose each one of up just typed in ONE program? That would mean at least 1000 programs we could all share! Of course there will always be those content to sit back and "Let Gary do it.." I guess there's pretty much nothing to be done about that. They will miss out on learning some BASIC. Oh well!

The last topic concerning Type-ins is the new CK-TYPE program that Stan Lemke recently wrote for TIME DESIGNS MAGAZINE (May/June 87). This aids the Type-in process in the following way. The original author of the Type-in must run his program through CK-TYPE and print that output along with the program listing itself. CK-TYPE will do a checksum of the characters in each BASIC line and list these. Then, when our intrepid typer goes to input the program, he ALSO needs CK-TYPE and runs his final input through the CK-TYPE as well. He will ALSO get a listing of checksum values which he compares with those of the original author. It's not perfect, but it's way ahead of nothing. For additional copies of this issue of TDM, send $3 (includes postage) to TIME DESIGNS/29722 Hult Rd/ Colton, OR 97017. If all newsletters used this system, it would greatly enhance the whole process. Other computer types have had such programs for years.

So the main advantages of PD software are these: A)It's Free, B) it helps YOU learn to program, C) It's easy to modify and D) there are some PD programs which can't be bought at ANY price.

PUBLIC DOMAIN CHART

Title	A	B	C	D	E
SPECTRUM PROGRAMS	3	68	22.7	?	68+
ZX COMPUTING	10	123	12.3	38	467
TIME DESIGNS	10	64	6.4	16	102
SINCLAIR USER	19	74	3.9	62	241
T/S HORIZONS	8	27	3.4	20	67
YOUR SPECTRUM	13	40	3.1	40	123
SYNCHWARE NEWS	10	30	3.0	23	69
SUM	12	25	2.1	41	85
LISTing	15	21	1.4	35	49
QL USER / WORLD	14	87	6.2	?	87+
		559			1358

Column A is the number of issues in the sample.
Column B is the total number of type-ins found in the entire sample.
Column C is the average number of Type-ins per issue.
Column D is the total number of issues known to exist (as of June 1987).
Column E is an extrapolation using the preceeding columns to give a "Ball park" figure of the total number of type-ins in existance.
The chart is in order of the most to least dense per issue.

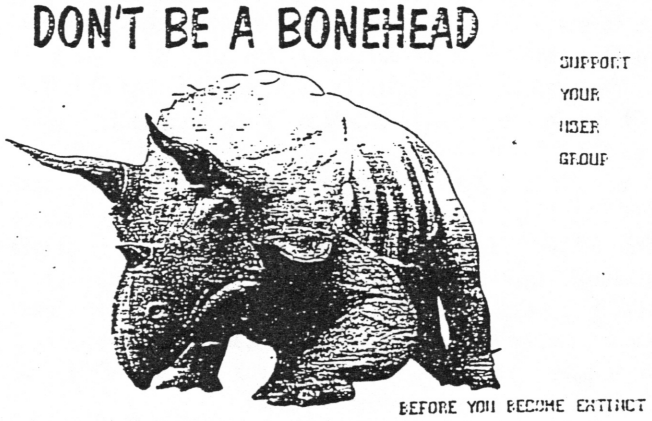
-=Shareware/Fairware=-

A fairly new phenomena which has emerged in recent years is the concept of Shareware, also called Fairware or Freeware. In this case, the program is freely exchanged among users, as long as it's accompanying "commercial message" remains intact. This message simply states,"If you liked this program and found it useful, please send $X.00 to the author at:(and gives the address). Many people, of course, never pay. They end up missing out on updates, corrections, bug "fixes" and good quality DOC's. This is the main advantage of Fairware over normal PD. It's a living, ongoing thing with support.

The most famous T/S Fairware author is Jack Dohaney. He calls his enterprise, "Jack's Fairware" and he's currently offering over 20 titles. You can get his latest catalog by sending a S.A.S.E. to Jack at: 390 Rutherford Av./Redwood City, CA 94061. My personal favorites are his MSCRIPT upgrade (which I'm using to write this) and the RLE Decoder/Encoder program called QRL-the best available at this time.

Jack describes Fairware this way,"FAIRWARE is..CONDITIONAL public-domain user-supported software that is NOT exactly free. FAIRWARE costs whatever the USER feels is a fair price for what he or she received.

"The MAIN IDEA behind Fairware is that support is a two- way street: user and author should support each other, for their mutual benefit...."



Binary Data Encoding

Computer data streams are nothing more than a sequence of electrical highs and lows (1's and 0's). Yet the ability to differentiate 1's from 0's allows us to code virtually anything. A simple eight-bit sequence of ones and zeros can produce any number from 0 – 255. Computer scientists assigned these 256 binary values to a corresponding character set. The character set includes all of the letters and numbers, punctuation, and a variety of control characters. As a result, every time a computer receives a binary 01000010 (decimal 66, hex 42), it interprets it as the character "B." Or, similarly, every time a computer receives a binary 00110001 (decimal 49, hex 31), it interprets it as a "1."

There are more than one of these code sets, but the most common is ASCII, or American Standard Code for Information Interchange. ASCII was developed by the American National Standards Institute and has attained nearly universal acceptance. EBCDIC or Extended Binary Coded Decimal Interchange Code has been used widely in IBM® applications. Baudot and transcode are leftovers from ticker tape and Telex® applications.

Accustomed to reading numbers in decimal, most people find eight-bit binary values difficult to read. So we usually look at the numeric representation of a character in hexadecimal (base 16) or decimal (base 10). Now, remember that binary numbers are comprised of 0's and 1's, and decimal numbers are comprised of numbers 0 through 9. To construct a hexadecimal (base 16) number, we need digits for 0 – 15. So we add A, B, C, D, E and F to the numbers 0 – 9. (A corresponds to decimal 10, B to 11, C to 12, D to 13, E to 14, and F to 15.) Then decimal 16 becomes 10 in hexadecimal. (Note that the 10's column in decimal has become the 16's column in hexadecimal.)

We have included the chart below to give you a complete cross reference between character code sets and their decimal and hexadecimal representations. Note that the ASCII hex value changes depending upon the parity selected. This is because the parity bit is appended to the end of the binary representation of the character and therefore changes the resulting binary and hexadecimal numbers.

Universal Code Chart/Hexidecimal Equivalency

* The listing in the Baudot may be in the form of two hex numbers, in which case the first number designates the LETTERS / FIGURE SHIFT. Where only one hex number is shown, the character is shift independent.

Character or Control	Decimal	HEX: Mark Parity ASCII	Space Parity ASCII	Even Parity ASCII	Odd Parity ASCII	EBCDIC	6-Bit Transcode	Baudot*
A	65	C1	41	41	C1	C1	01	1F 03
B	66	C2	42	42	C2	C2	02	1F 19
C	67	C3	43	C3	43	C3	03	1F 0E
D	68	C4	44	44	C4	C4	04	1F 09
E	69	C5	45	C5	45	C5	05	1F 01
F	70	C6	46	C6	46	C6	06	1F 0D
G	71	C7	47	47	C7	C7	07	1F 1A
H	72	C8	48	48	C8	C8	08	1F 14
I	73	C9	49	C9	49	C9	39	1F 06
J	74	CA	4A	CA	4A	D1	11	1F 0B
K	75	CB	4B	4B	CB	D2	12	1F 0F
L	76	CC	4C	CC	4C	D3	13	1F 12
M	77	CD	4D	4D	CD	D4	14	1F 1C
N	78	CE	4E	4E	CE	D5	15	1F 0C
O	79	CF	4F	CF	4F	D6	16	1F 18
P	80	D0	50	50	D0	D7	17	1F 16
Q	81	D1	51	D1	51	D8	18	1F 17
R	82	D2	52	D2	52	D9	19	1F 0A
S	83	D3	53	53	D3	E2	22	1F 05
T	84	D4	54	D4	54	E3	23	1F 10
U	85	D5	55	55	D5	E4	24	1F 07
V	86	D6	56	56	D6	E5	25	1F 1E
W	87	D7	57	D7	57	E6	26	1F 13
X	88	D8	58	D8	58	E7	27	1F 1D
Y	89	D9	59	59	D9	E8	28	1F 15
Z	90	DA	5A	5A	DA	E9	29	1F 11
a	97	E1	61	E1	61	81		
b	98	E2	62	E2	62	82		
c	99	E3	63	63	E3	83		
d	100	E4	64	E4	64	84		
e	101	E5	65	65	E5	85		

Character or Control	Decimal	HEX: Mark Parity ASCII	Space Parity ASCII	Even Parity ASCII	Odd Parity ASCII	EBCDIC	6-Bit Transcode	Baudot*
f	102	E6	66	66	E6	86		
g	103	E7	67	E7	67	87		
h	104	E8	68	E8	68	88		
i	105	E9	69	69	E9	89		
j	106	EA	6A	6A	EA	91		
k	107	EB	6B	EB	6B	92		
l	108	EC	6C	6C	EC	93		
m	109	ED	6D	ED	6D	94		
n	110	EE	6E	EE	6E	95		
o	111	EF	6F	6F	EF	96		
p	112	F0	70	F0	70	97		
q	113	F1	71	71	F1	98		
r	114	F2	72	72	F2	99		
s	115	F3	73	Fc	73	A2		
t	116	F4	74	74	F4	A3		
u	117	F5	75	F5	75	A4		
v	118	F6	76	F6	75	A5		
w	119	F7	77	77	F7	A6		
x	120	F8	78	78	F8	A7		
y	121	F9	79	F9	79	A8		
z	122	FA	7A	FA	7A	A9		
0	48	B0	30	30	B0	F0	30	1B 16
1	49	B1	31	B1	31	F1	31	1B 17
2	50	B2	32	B2	32	F2	32	1B 13
3	51	B3	33	33	B3	F3	33	1B 01
4	52	B4	34	B4	34	F4	34	1B 0A
5	53	B5	35	35	B5	F5	35	1B 10
6	54	B6	36	36	B6	F6	36	1B 15
7	55	B7	37	B7	37	F7	37	1B 07
8	56	B8	38	B8	38	F8	38	1B 06
9	57	B9	39	39	B9	F9	39	1B 18

Character or Control	Decimal	HEX: Mark Parity ASCII	Space Parity ASCII	Even Parity ASCII	Odd Parity ASCII	EBCDIC	6-Bit Transcode	Baudot*
SP	32	A0	20	A0	20	40	1A	04
!	33	A1	21	21	A1	5A		
"	34	A2	22	22	A2	7F		1B 11
#	35	A3	23	A3	23	7B	3B	1B 14
$	36	A4	24	24	A4	5B	1B	1B 00
%	37	A5	25	A5	25	6C	2C	
&	38	A6	26	A6	26	50	10	1B 1A
'	39	A7	27	27	A7	7D		1B 05
(	40	A8	28	28	A8	4D		1B 0F
)	41	A9	29	A9	29	5D		1B 12
*	42	AA	2A	AA	2A	5C	1C	
+	43	AB	2B	2B	AB	4E		1B 09
,	44	AC	2C	AC	2C	6B	2B	1B 0C
-	45	AD	2D	2D	AD	60	20	1B 03
.	46	AE	2E	2E	AE	4B	0B	1B 1C
/	47	AF	2F	AF	2F	61	21	1B 1D
:	58	BA	3A	3A	BA	7A		1B 0E
;	59	BB	3B	BB	3B	5E		1B 1E
<	60	BC	3C	3C	BC	4C	0C	
=	61	BD	3D	BD	3D	7E		
>	62	BE	3E	BE	3E	6E		
?	63	BF	3F	3F	BF	6F		1B 19
@	64	C0	40	C0	40	7C	3C	
[	91	DB	5B	DB	5B			
\	92	DC	5C	5C	DC	E0		
]	93	DD	5D	DD	5D			
^	94	DE	5E	DE	5E			
_	95	DF	5F	5F	DF	6D		
`	96	E0	60	60	E0			
{	123	FB	7B	7B	FB	C0		
|	124	FC	7C	FC	7C	6A		
}	125	FD	7D	7D	FD	D0		

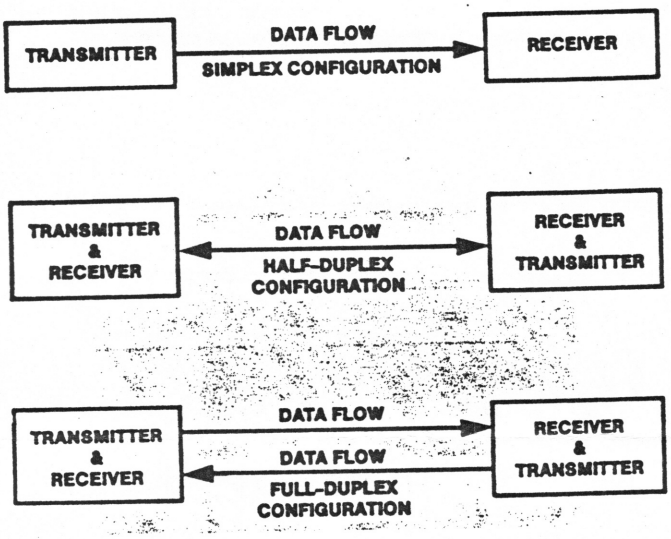
Character or Control	Decimal	HEX: Mark Parity ASCII	Space Parity ASCII	Even Parity ASCII	Odd Parity ASCII	EBCDIC	6-Bit Transcode	Baudot*
~	126	FE	7E	7E	FE	A1		
¢	155		9B			4A		
¬	191		BF			5F		
						4F		
ACK	6	86	06	06	86	2E		
BEL	7	87	07	87	07	2F	00	1B 08
BS	8	88	08	88	08	16	•	
BYP						24		
CAN	24	98	18	18	98	18		
CC						:A		
CR	13	80	0D	80	0D	00		08
DC1	17	91	11	11	91	11		
DC2	18	92	12	12	92	12		
DC3	19	93	13	93	13	13		
DC4	20	94	14	14	94	3C		
DEL	127	FF	7F	FF	7F	07	3F	
DLE	16	90	10	90	10	10	1F	
DS						20		
EM	25	99	19	99	19	19	3E	
ENQ	5	85	05	05	85	2D	2D	
EOB						26		
EOT	4	84	04	84	04	37	1E	
ESC	27	98	1B	1B	98	27	2A	
ETB	23	94	17	17	97	26	0F	
ETX	3	83	03	03	83	03	2E	
FF	12	8C	0C	0C	8C	0C		
FS	28	9C	1C	9C	1C	22		
GS	29	90	1D	1D	90			
HT	9	89	09	09	89	05	2F	
IFS						1C		
IGS						1D		

Character or Control	Decimal	HEX: Mark Parity ASCII	Space Parity ASCII	Even Parity ASCII	Odd Parity ASCII	EBCDIC	6-Bit Transcode	Baudot*
IL						17		
IRS						1E		
IUS						1F		
LC						06		
LET								1F
LF	10	8A	0A	0A	8A	25		02
NAK	21	95	15	95	15	3D	3D	
NL						3D	3D	
NUL	0	80	00	00	80	00		
PF						04		
PN						34		
PRE						27		
RES						14		
RLF						09		
RS	30	9E	1E	1E	9E	35		
SI	15	8F	0F	0F	8F	0F		
SM						2A		
SMM						0A		
SO	14	8E	0E	8E	0E	0E		
SOH	1	81	01	81	01	01	00	
SOS						21		
STX	2	82	02	82	02	02	0A	
SUB	26	9A	1A	9A	1A	3F	0E	
SYN	22	96	16	96	16	32	3A	
UC						36		
US	31	9F	1F	9F	1F		1D	
VT	11	88	0B	88	0B	08		

Serial, Asynchronous Data Transmission

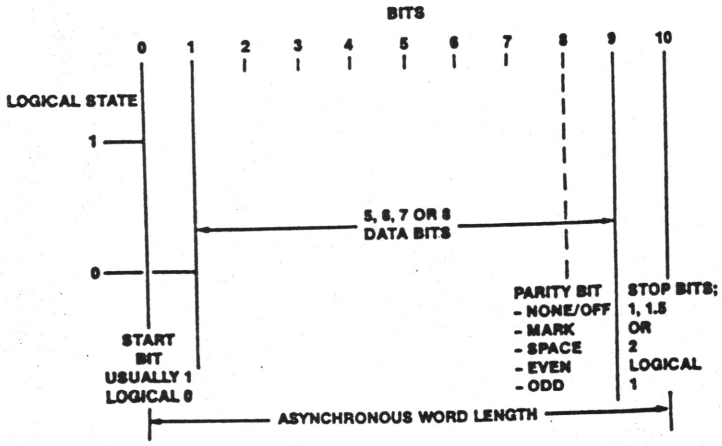
General

In serial transmission, information is transferred, one data bit at a time, between two computing devices. The flow of this data can follow one of three transmission modes: simplex, half duplex or full duplex. Simplex allows data flow in only one direction; half duplex allows data flow in both directions (but not simultaneously); while full duplex allows simultaneous two-way transmission.



Asynchronous Data Frame

Within the asynchronous data stream, each character of data is transported in a binary bit frame. Each frame begins with a start bit. A low-voltage signal on the data line marks the beginning of the start bit, and the receiving device can then begin looking for 0's and 1's. The following five to eight data bits (the number depends on the code used) comprise the binary character. For error detection, an optional parity bit can mark whether the total of 0's or 1's was even or odd. A stop bit signals the end of the character. (Stop bits range in length from one to two bits. The slower the speed of transmission, the more stop bits required for end-of-frame recognition.)


BITS
0 1 2 3 4 5 6 7 8 9 10
LOGICAL STATE
1
0
5, 6, 7 OR 8 DATA BITS
PARITY BIT - NONE/OFF - MARK - SPACE - EVEN - ODD
STOP BITS; 1, 1.5 OR 2 LOGICAL 1
START BIT USUALLY 1 LOGICAL 0
ASYNCHRONOUS WORD LENGTH

Parity bits trap errors in the following manner: when the transmitting device frames a character, it tallies the number of 0's or 1's within the frame and attaches a parity bit. (The parity bit will vary according to whether the total is even or odd.) Then, the receiving end will count the 0's or 1's and compare the total to the "odd" or "even" recorded on the parity bit. If the receiving end finds a discrepancy, it can flag the data and request a retransmission.

Parity selections other than odd and even include mark, space and off. Mark always sets parity at 1; space sets parity at 0; and off tells the system to ignore the parity bit.

PARITY TYPE	DESCRIPTION
ODD	EIGHTH DATA BIT IS LOGICAL ZERO IF TOTAL NUMBER OF LOGICAL 1's IN FIRST SEVEN DATA BIT IS ODD
EVEN	EIGHTH DATA BIT IS LOGICAL ZERO IF TOTAL NUMBER OF LOGICAL 1's IN FIRST SEVEN DATA BIT IS EVEN
MARK	EIGHTH DATA BIT ALWAYS LOGICAL 1 (HIGH/MARK)
SPACE	EIGHTH DATA BIT ALWAYS LOGICAL 0 (LOW/SPACE)
NONE/OFF	EIGHTH DATA BIT IGNORED

Asynchronous Link Control

We have defined the asynchronous frame and the directional modes, but we haven't covered the rules for orderly transmission. Each end of the link has to conform to certain operating parameters in order for flow to occur without loss of data.

The first parameter is the rate of transmission. Serial data transmission is measured in bits per second (bps). Typical asynchronous serial baud rates are 110, 300, 600, 1200 and 2400 bps. In order for two computing devices to interact, both must be operating (transmitting or receiving data) at the same baud rate, or there must be an intermediate memory device, called a buffer, that accommodates the differences in speed.

In addition to transmit and receive buffers, most serial asynchronous links use a flow control system to handle data transmission. A common asynchronous serial flow control is X-ON/X-OFF. When a receive buffer reaches its memory capacity, the receiving device sends an ASCII DC3 (X-OFF) signal to the transmitting device telling it to stop sending data. When the receive buffer has sufficiently unloaded, it transmits an ASCII DC1 (X-ON) signal back to the transmitting device, telling it to continue transmission.

Other flow controls commonly used in asynchronous communications are lead control and the ENQ/ACK protocol. A lead control protocol controls data flow by alternately raising and lowering the voltage on a lead of the RS-232 interface. DTR (pin 20) and CTS (pin 5) are often used. When a pin's voltage is high, data flow is enabled; when it drops low, data flow is stopped.

ENQ/ACK is a block-oriented protocol, i.e. a protocol that sends a fixed amount of characters every time it transmits. Typically, the transmitting device sends an ENQ character and waits for an ACK character from the receiving device before beginning transmission. Once the ACK character is received, the entire block is transmitted. When the block transmission is complete, the transmitting device once again begins polling the receiving device with ENQs.

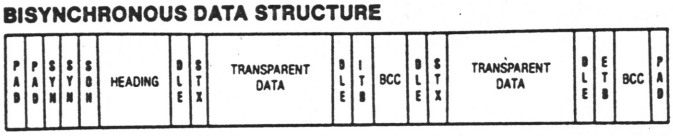
A Few Words about Synchronous Data Transmission

In Synchronous Data Transmission, special characters synchronize the transmitting and receiving elements of the link; this permits transmissions to occur without the overhead of start and stop bits as in asynchronous communication.

Synchronous protocols fall into two major categories: 1) character- or byte-oriented protocols that specify a definite character length, and 2) bit-oriented protocols that do not specify character boundaries.

Bisynchronous (BSC), short for Binary Synchronous Communication, is one of the most common character-oriented protocols. Bisynchronous uses a set of special characters to define the structure of the data transmission frame. At the start of each block of data are PAD and SYN characters which signal the start of a frame and allow the receiving station to synchronize with the transmitting station clock. A variety of different bisynchronous transmission frames can be created using the available character control set. Below we show the list of characters and a typical bisynchronous frame:

BISYNCHRONOUS DATA STRUCTURE

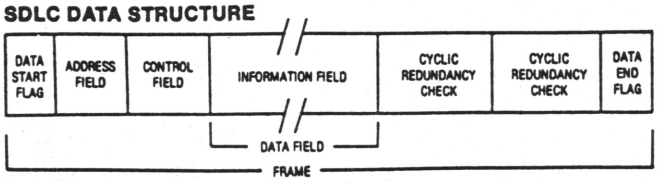


BISYNCHRONOUS CONTROL CHARACTERS

BISYNCHRONOUS CHARACTER	HEX VALUE	CHARACTER DESCRIPTION
SYN	32	Synchronous idle
PAD	55	Start of frame pad
PAD	FF	End of frame pad
DLE	10	Data line escape
ENQ	2D	Enquiry
SOH	01	Start of heading
STX	02	Start of text
ITB	1F	End of intermediate block
ETB	26	End of transmission (block)
ETX	03	End of text

SDLC, short for Synchronous Data Link Control, typifies the second type of synchronous protocol — the bit-oriented protocol. Instead of using a control character set as does bisynchronous, SDLC uses a variety of bit patterns to flag the beginning and end of a frame. Other bit patterns are used for the address, control and packet header fields which route the frame through a network to its destination. Below we show a typical SDLC transmission frame:

SDLC DATA STRUCTURE



Please note that a complete description of synchronous data transmission is out of the scope of this limited space. We recommend the following for further reference:

IBM® Publications:

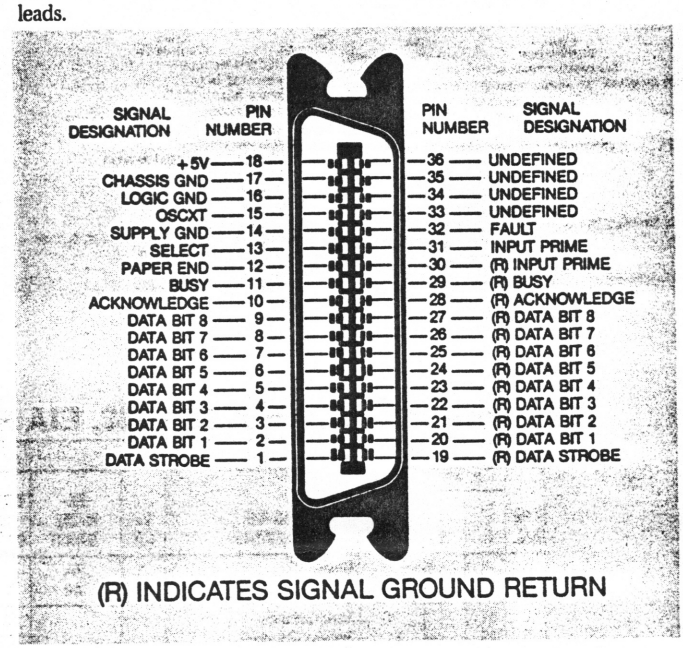
Binary Synchronous Communications — IBM Order No. 6A27-3004-4

Synchronous Data Link Control — IBM Order No. 6A27-3093-2

Centronics® Parallel Interface

The Centronics Parallel Interface is a 36-pin, byte-wide interface that has become a widely accepted standard for computer-to-printer communications. The interface has eight lines which carry their respective binary bits in parallel. The transmission of these data bits is controlled by the computer-supplied STROBE pulse. Handshaking (flow control) is achieved by asserting or deasserting either the ACKNLG or BUSY leads or both. All Centronics Parallel logic levels are TTL.

For your convenient reference, we have included a diagram of the Centronics connector and a chart that covers the most commonly used leads.



SIGNAL PIN NO.	RETURN PIN NO.	SIGNAL	DIRECTION (with ref. to printer)	DESCRIPTION
1	19	STROBE	In	STROBE pulse (negative going) enables reading data.
2	20	DATA 1	In	1st to 8th bits of parallel data. Each signal is at "HIGH" level when data is logical "1" and "LOW" when logical "0".
3	21	DATA 2	In	
4	22	DATA 3	In	
5	23	DATA 4	In	
6	24	DATA 5	In	
7	25	DATA 6	In	
8	26	DATA 7	In	
9	27	DATA 8	In	
10	28	ACKNLG	Out	"LOW" indicates that data has been received and that the printer is ready to accept other data.
11	29	BUSY	Out	"HIGH" indicates that the printer cannot receive data.

NOTE: Pins 12, 13, 14, 15, 18, 31, 32, 34, 35 and 36 vary in function depending upon application; they are commonly used for printer auxiliary controls, and error handling and indication.

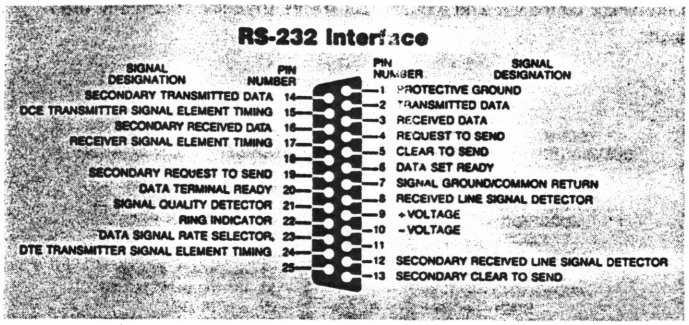
Pins 16 and 17 are commonly used for logic ground and chassis ground, respectively.

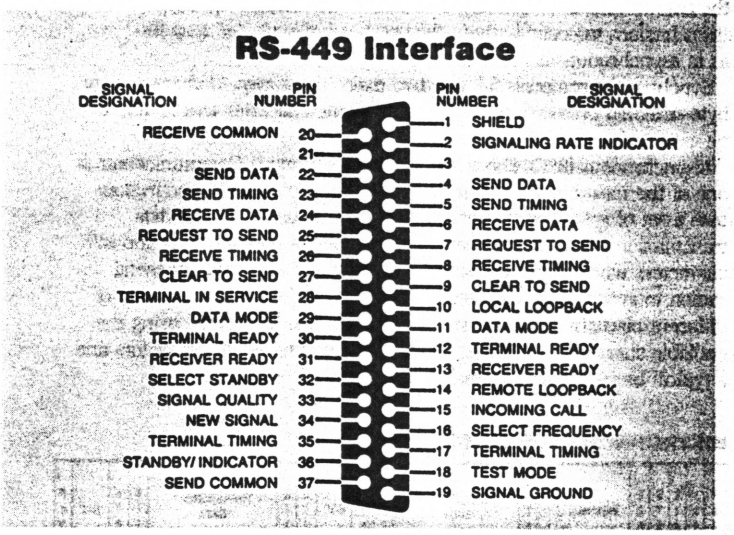
RS–232 and RS–449 Interfaces

RS–232 is an Electronic Industries Association (EIA) standard, applicable to the 25–pin interconnection of Data Terminal Equipment (DTE) and Data Communication Equipment (DCE) employing serial binary data interchange.

RS–449 is an EIA standard, applicable to the 37–pin and 9–pin interconnection of DTE and DCE devices that use the serial binary data interchange.

Both of these standards have been widely accepted in the field of computer and data communications. We have included the following diagrams and chart for convenient reference.





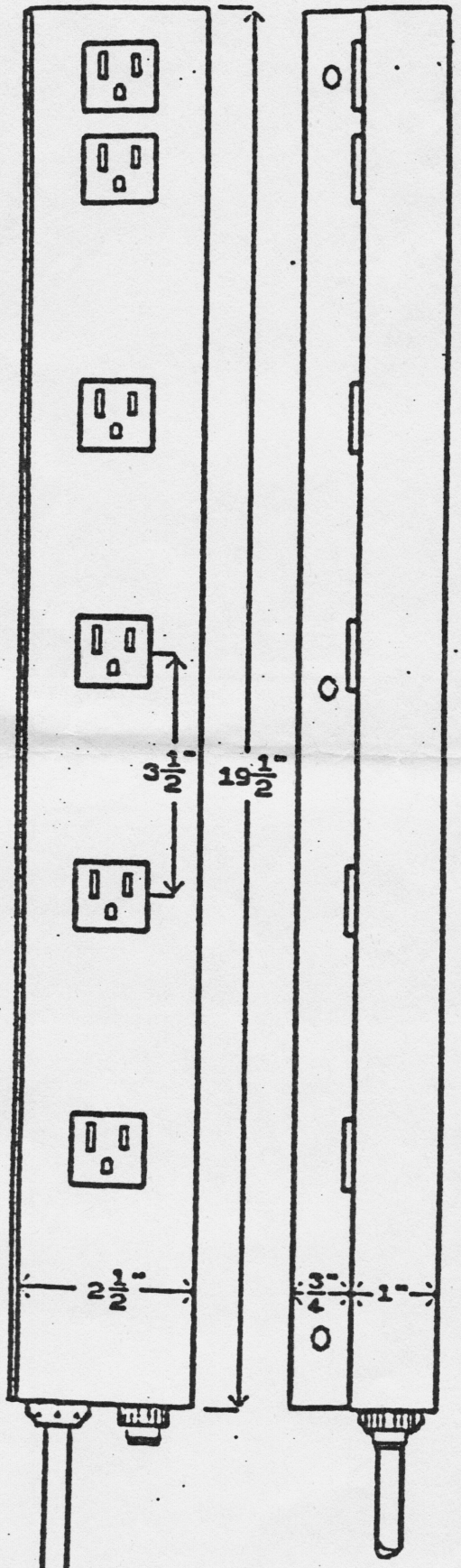
Pinout Table For EIA RS–449, EIA RS–232C/CCITT V.24

RS–449 Interface					RS–232 Interface				Signal Type and Direction						
	37 PIN								GND	DATA		CONTROL		TIMING	
9 PIN AUX.	A	B	RS–449 CIRCUIT	RS–449 DESCRIPTION	25 PIN	EIA–RS–232 CIRCUIT	CCITT–V.24 CIRCUIT	RS–232 DESCRIPTION		From DCE	To DCE	From DCE	To DCE	From DCE	To DCE
1	1			Shield	1	AA	101	Protective Ground	X						
5	19		SG	Signal Ground	7	AB	102	Signal Ground/Common Return	X						
9	37		SC	Send Common			102a	DTE Common	X						
6	20		RC	Receive Common			102b	DCE Common	X						
	4	22	SD	Send Data	2	BA	103	Transmitted Data			X				
	6	24	RD	Receive Data	3	BB	104	Received Data		X					
	7	25	RS	Request to Send	4	CA	105	Request to Send					X		
	9	27	CS	Clear to Send	5	CB	106	Clear to Send				X			
	11	29	DM	Data Mode	6	CC	107	Data Set Ready				X			
	12	30	TR	Terminal Ready	20	CD	108.2	Data Terminal Ready					X		
	15		IC	Incoming Call	22	CE	125	Ring Indicator				X			
	13	31	RR	Receiver Ready	8	CF	109	Received Line Signal Detector				X			
	33		SQ	Signal Quality	21	CG	110	Signal Quality Detector				X			
	16		SR	Signaling Rate Selector	23	CH	111	Data Signal Rate Selector (DTE)					X		
	2		SI	Signaling Rate Indicator	23	CI	112	Data Signal Rate Selector (DCE)				X			
	17	35	TT	Terminal Timing	24	DA	113	Transmitter Signal Element Timing (DTE)							X
	5	23	ST	Send Timing	15	DB	114	Transmitter Signal Element Timing (DCE)						X	
	8	26	RT	Receive Timing	17	DD	115	Receiver Signal Element Timing (DCE)						X	
3			SSD	Secondary Send Data	14	SBA	118	Secondary Transmitted Data			X				
4			SRD	Secondary Receive Data	16	SBB	119	Secondary Received Data		X					
7			SRS	Secondary Request to Send	19	SCA	120	Secondary Request to Send					X		
8			SCS	Secondary Clear to Send	13	SCB	121	Secondary Clear to Send				X			
2			SRR	Secondary Receiver Ready	12	SCF	122	Secondary Received Line Signal Detector				X			
	10		LL	Local Loopback			141	Local Loopback					X		
	14		RL	Remote Loopback			140	Remote Loopback					X		
	18		TM	Test Mode			142	Test Indicator				X			
	32		SS	Select Standby			116	Select Standby					X		
	36		SB	Standby Indicator			117	Standby Indicator				X			
	16		SF	Select Frequency			126	Select Transmit Frequency				X			
	28		IS	Terminal in Service									X		
	34		NS	New Signal									X		

NOTE: DB25 references the (25–pin) connector that is commonly used for RS–232 and V.24. Pins 9 and 10 are reserved for data set testing. Pins 11, 18 and 25 are undefined. Pins 3 and 21 of RS–449 interface connector are undefined. Lead 23 of RS–232 connector may be defined as CH or CI. The 37–pin designation B equals return.

Tired of tackling "WHERE TO PUT THE POWER PACK" problems??

....maybe covering an adjacent outlet?
....or hanging it off the side of the power strip?
If you don't mind these problems, then you don't need the



"LINEPACKER"

But if you want a better solution for this nagging problem, then here it is! A power strip with room to plug in all those oversize power packs.
The LINEPACKER allows up to five power packs to be plugged in without loss of outlets or pack "hangover". Even the very large Sinclair 2068 computer and 2040 printer packs can be used with no wasted outlets.

LINEPACKER's features include:

* 6 grounded outlets - 5 spaced 3-1/2" from center to center.
* 15 ampere circuit breaker.
* MOV protected outlets (Metal Oxide Varistor)
* 8 foot heavy duty 12 gauge stranded copper power cord
* reversible mounting flange
* low profile 1-3/4"H x 2-1/2"W x 19-1/2"L
* heavy gauge steel construction

LINEPACKER's best feature is the price!

$19.95 +ship

Quantities are limited. Units are surplus equipment, some have scuffs or scratches but still in good working condition. Color is black only.

To ORDER send personal check, cashiers check, or money order for $23.00 each (INCLUDES $3.05 postage and handling)

To:

BOB CRECCO
1627 DEWEY STREET
NEW ALBANY, INDIANA
47150

S.L.U.G.
4122 Wallingford Ln.
Louisville, Ky.
40218 2365
USA

DON LAMBERT
3310 Clover Dr. S.W.
CEDAR RAPIDS, IOWA
52404